High Level Organization of Safety Arguments

S.A. Bates; University of York; York, UK

I.J. Bate; University of York; York, UK

J.A. McDermid; University of York; York, UK

Keywords: Safety Cases, Safety Case Architectures, System Safety

Abstract

The work presented in this paper has been conducted as part of the BAE Systems funded Dependable Computing Systems Centre (DCSC). In this paper we build on previous Safety Case Architecture (SCA) work and introduce a technique called **Imaginative Anticipation** and two quality characteristics to aid the development of a SCA.

SCAs are used, in this paper, to facilitate the high level organisation of the safety case into argument modules. A safety case presents a reasoned argument that uses identified evidence to justify the safety of a particular system in a particular context. Therefore, the arrangement into argument modules is conducted to develop modular safety cases. In this paper we discuss reasons for why a modular safety case would be developed and how a SCA can be used to support the development.

A process is introduced for creating SCAs. The process consists of four iterative phases: Knowledge gathering, proposition of SCAs, evaluation of proposed SCAs, and, selection of the most appropriate SCA. The four phases are discussed and the process is demonstrated through a small example.

Introduction

The work presented in this paper has been conducted as part of the BAE Systems funded Dependable Computing Systems Centre (DCSC). The DCSC is a research project running in the High Integrity Systems Engineering (HISE) research group at the University of York, UK. The DCSC has been in existence for 13 years and has contributed much to the safety engineering domain. Current DCSC research is investigating the use of object oriented software development techniques for safety related systems, and modular safety cases. In this paper we present some of the results from the continuing investigation into modular safety cases.

GSN is an argument notation developed by the HISE group. It is used in this paper for the representation of SCAs because of its support for argument modularity. The notation includes various symbols facilitating the graphical representation of non-modular and modular safety arguments. GSN has been widely adopted by the safety-critical domain to aid the construction of safety cases. Furthermore, software tool support for GSN is available from several vendors.

Modular safety cases are a proposed approach to improving change management of safety cases. A safety case presents a reasoned argument that uses identified evidence to justify the safety of a particular system in a particular context. An argument module contains related elements of argument and evidence, and is as independent as possible of other argument modules. Therefore, a modular safety case shows how argument modules can be combined to provide justification for the safety of a system in a particular context.

In this paper we use a Safety Case Architecture (SCA) for the high level organisation of the safety case into argument modules. We develop a SCA using an example of a computer based safety-related system. It is assumed for the purposes of this paper that such a system consists of software and a target hardware platform, other aspects such as networks, sensors, actuators, etc., are not considered. It is then shown how the SCA can then be used to construct a modular safety case.

The contribution of this paper is to the creation and evaluation of SCAs. In this paper we introduce a process for the creation of SCAs and two quality characteristics (acceptability and changeability) for the evaluation of SCAs. It is then shown how the quality characteristics (QCs) can be used to evaluate proposed SCA during the evaluation aspect

of the proposed four phase iterative process.

The Goal Structuring Notation and Modularity

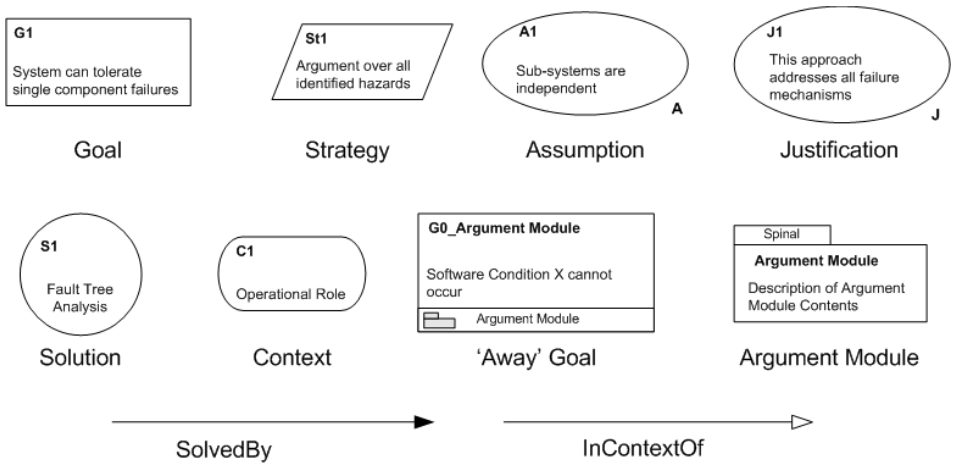

Figure 1 — GSN Symbols

The foundations for the SCA research have been derived from GSN and the modular features of GSN. The GSN symbols are shown in figure 1 and can be classified into the following four categories:

- Argument Symbols - 'Goal', 'Strategy', and 'Solution' symbols
- Rationale Symbols - 'Context', 'Assumption', and 'Justification' symbols
- Linking Symbols - 'SolvedBy' and 'InContextOf' arrows
- Modular Symbols - 'Argument Module' and 'Away Goal' symbols

The argument symbols are used to show how the claims (**Goal** symbol) being made about the system are decomposed into sub-claims until a direct reference to available evidence (**Solution** symbol) can be made. Further clarification of the approach(es) taken can be provided by making explicit the **Strategy** taken to construct the argument. Argument symbols are linked using the **SolvedBy** arrow. The rationale symbols are used to provide an explanation for the stated argument. **Context** is used to link the argument symbols to information essential to their interpretation, e.g., definitions, system models, etc. **Assumption** symbols are used to link argument symbols to information necessary to understand their validity. **Justification** symbols are used to link argument symbols to reasons why they are stated in particular manner. Rationale symbols are linked to the relevant argument symbols using the **InContextOf** arrow.

The modular symbols are a special case as they can be used as either an argument symbol or a rationale symbol. When an **Argument Module** symbol is used as an argument symbol, it indicates that a large-scale body of argument and evidence supports the claim or strategy it is linked to. When using the **Away Goal** symbol as an argument symbol, it indicates a direct link (see fig. 2) to part of an argument contained within an argument module, which provides support for the claim or strategy it is linked to. When using the modular symbols as context they indicate that the whole (**Argument Module**) or part (**Away Goal**) of an argument contained within an argument module is providing rationale for the argument, e.g., a goal "All Identified Hazards Have Been Addressed In Accordance With The ALARP Principle", could be contextually linked to an argument module in which arguments about the rigorousness of the hazard identification process are contained.

Safety Cases and Safety Arguments

Safety Cases, in some form, are required by several safety standards. In reference 7 a safety case consists of: factual information about the system and its subsystem, their functions and interfaces; safety arguments, which present a

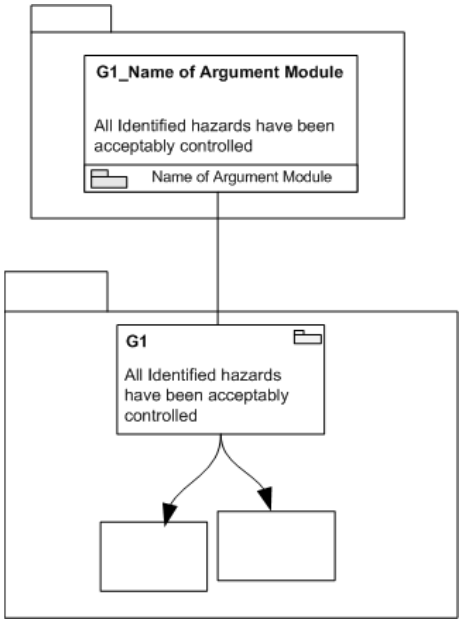

Figure 2 — Figure Showing A Direct Link Between Argument Modules

reasoned account that uses available evidence to justify why a particular system is acceptably safe to operate; a description of the means employed to prevent an identified hazard from occurring. Therefore, a safety case can be defined as:

> *A safety case should communicate a clear, comprehensible, and defensible argument* [using available evidence] *that a system is acceptably safe to operate in a particular context* (ref. 3)

Insight into constructing a safety case can is given here from the argumentation domain. In reference 10 several types of argument dialogue are identified. Safety cases fit in two of the identified types of dialogue:

- **Inquiry dialogue**, where the goal is to (dis)prove a claim
- **Persuasion dialogue**, where the goal is resolve or clarify issues

Safety cases most strongly fit into the inquiry dialogue because the aim of the safety case is to (dis)prove claims, e.g., that a system is acceptably safe to operate in a particular context. This aim is achieved by gathering evidence that will (dis)prove the validity of a claim. For safety cases there are several sources for this evidence, which include: the design, the development process, the safety process, simulations, field experience etc. The evidence is connected to claims using safety arguments that can be constructed using GSN.

Safety cases fit into the persuasion dialogue because of the need to clarify the reasons for making particular safety arguments. The clarification (i.e, context, assumption, and justification) can be used to resolve possible disagreements or rebuttals that other parties (e.g., customers) have with the safety arguments. This can be done by stating what is considered to be within and outside of the scope of the safety case. The result is that the safety case can be used as a persuasion tool. That is, based on the clarification of the safety arguments, the rebuttals and disagreements of other parties have been resolved and the claims made about a system can be reasonably accepted.

Modular safety cases have been recognised as a “step in the right direction” for safety argument construction (refs. 4, 9). A modular safety case is developed by decomposing the safety arguments and evidence into argument modules. This modular decomposition should minimise the potential web of dependencies usually experienced in monolithic safety case construction by facilitating the independent development of argument contained in argument modules. This approach is recommended as it should improve the potential changeability compared to a non-modular (or monolithic)

safety case. The improved changeability arises from the independence of the argument modules, i.e, it should be possible to change the arguments in one module independently of other modules.

The modular safety case can also improve the previously mentioned inquiry and persuasion aspects of the safety case. By presenting the high level (or modular) organisation the inquiry aspect of the safety case can be improved by allowing better association of the available evidence to the argument modules, and, as a working hypothesis, support modularisation of evidence. The persuasion aspect of the safety case is also improved because the high level organisation can be used to clarify the scope of, and approach advocated by, the modular safety case. This clarification is achieved by allowing the overall scope and structure of the safety case to be more visually understood.

The High Level Organisation of the Safety Case

In reference 4 Safety Case Architectures (SCAs) were introduced to facilitate the high level organisation of the safety case. In this paper we identify that SCAs are useful for the development of modular safety cases and hence we define SCA in following terms:

A Safety Case Architecture is the master plan for a particular modular safety case

The master plan should provide safety case developers, maintainers, assessors, etc., (these will be referred to collectively as SCA users) with at least the following:

- Enough information to ensure that any interested party can easily develop, locate, maintain, assess, etc., (these will be referred to collectively as SCA uses) the arguments expected to be contained in the modular safety case
- The long term plan for the modular safety case

There are several aspects to ensuring that the SCA provides enough information to the SCA users. Firstly, the SCA should present a graph, using argument modules, showing the master plan. This graph should provide SCA users wit sufficient information for them to use it, e.g., developers should be able to comprehend what they have to do to construct the modular safety case. Secondly, the SCA is decomposed until no new information or concept is introduced. This means that the architecture stops when it presents a compelling graph of the modular safety case to be produced, e.g., where the master plan indicates arguments about individual physical properties of a target hardware platform then the addition of further concepts will not add value to the SCA, i.e., the SCA presents enough information to compel the user. Other information include linking SCA argument modules to their directly related and developed (or instantiated) argument in the modular safety case, e.g., where the SCA indicates a argument module relating to {App X}[1] then "information" would be expected to link this to the relevant argument modules in the modular safety case providing independent argument about all the individual applications requiring safety argument in the safety case.

The long term aspect of the master plan was identified because of the need to support both modular and through-life use of a particular safety case (ref. 9). This aspect of SCA development is dependent on establishing the predicted changes to be made to as system a system (i.e., hardware platform upgrade) or the context surrounding the system (i.e, change in operating environment) and using these to aid the development of a SCA. The result should be a SCA that will maintain its integrity and limit the impact a change has on the overall safety case.

Creating a Safety Case Architecture

This section will introduce the four phase iterative process for creating a SCA. The proposed process consists of the following phases:

- Phase 1 – Knowledge Gathering
- Phase 2 – Proposition of potential SCA solutions using **Imaginative Anticipation**
- Phase 3 – Evaluation of proposed SCAs using Quality Characteristics (QCs)

[1] Curly braces are used to indicate a generic term to be instantiated

- Phase 4 – Selection of SCA

Knowledge Gathering:

Phase 1 of the creation of a SCA has been identified because in the domain of building architecture knowledge gathering is recognised as a important prerequisite activity to be performed before even "putting pen to paper" to start designing (ref. 6). This knowledge comes from several areas, which can be classified into knowledge of building and knowledge of environmental factors or context influencing the building design.

By analogy, 'knowledge of building' is the prerequisite information relating to structuring a SCA. In reference 5 the knowledge of structuring approaches is classified as architectural styles for a SCA. These styles include things such as decomposition around identified hazards or around system functions. This knowledge is essentially generic, but is required so that SCA can suggest appropriate SCAs in phase 2.

By analogy, 'knowledge of context' relates to a particular SCA to be developed. This knowledge comes from several areas including: the system design, standards to be addressed, the safety management plan, etc. Perhaps the most important contextual knowledge is of the predicted changes (i.e, maintenance and modifications) a particular system is likely to experience throughout its life. This knowledge is important because it will be used in phase 3 to aid the development of the long term master plan.

The knowledge gathering phase should provide a SCA developer with enough relevant information such that they have a clear picture of the SCAs they will suggest in phase 2.

SCA Proposition Using Imaginative Anticipation:

To create a SCA a proposed technique entitled **Imaginative Anticipation** is being developed. Imaginative anticipation considers two aspects of SCA development: (1) the argument modules found in the SCA and (2) the safety case patterns that could be used within the anticipated argument modules.

Imaginatively anticipating the modules found in a SCA is done by selecting the most appropriate decomposition approaches (i.e., around functions, hazards, system components, etc.,) indicates by phase 1. From these approaches possible SCAs are suggested.

Imaginatively anticipating the possible argument contained within the argument modules using safety case patterns is done to give further guidance to developers of argument modules. Safety case patterns are generalized or commonly experienced safety arguments[2]. The use of patterns aids the creation process by allowing SCA developers to imaginatively anticipate relationships that may exist between argument modules in the modular safety case, and therefore allows better evaluation in phase 3.

During this phase it may become apparent that insufficient or conflicting knowledge was gathered during phase 1, thus it may be necessary to repeat phase 1 to gather more knowledge or resolve conflicts.

Evaluating SCAs Using Quality Characteristics:

Phase 3 is where the suggested SCAs are evaluated using Quality Characteristics (QCs). Two primary QCs, **Acceptability** and **Changeability**, are introduced in this paper as a means of evaluating a SCA. These have been developed from research into the use of quality in several areas, the main sources being references 1, 4, 9, 2. The QC acceptability has been defined as:

> A judgement of how acceptable a particular SCA is in its current state and form

and changeability as:

> A judgement of how maintainable and modifiable a particular SCA is in its current state and form

[2] a discussion of patterns is out of the scope of this paper for further details refer to reference 3

The acceptability QC is introduced to ensure that judgements are made about how **Compelling** and **Comprehensible** a proposed SCA is. Compelling means that the modular safety case constructed from a particular SCA will present sufficient plausible safety arguments and as mentioned in the previous section no further concepts are required to make the SCA acceptable to users. Comprehensible means that any user is provided with enough information to understand how to use the SCA.

The changeability QC was identified for reasons including the need for easier maintenance and modifications of assurance cases. This QC can be further decomposed into the two sub QCs: Maintainability and Modifiability. Therefore changeability is used to influence development by evaluating a SCA for its support of maintenance and modifiability. This can be done using scenarios 2. Scenarios are used, in this case, to anticipate changes that may occur throughout the life of a SCA. The scenarios are based knowledge of the predicted system and context changes gathered during phase 1.

The maintainability sub-QC is used to ensure, as much as possible, that the modular safety case produced from a particular SCA will facilitate better change management of small-scale challenges than a traditionally constructed safety case, i.e., a change that has a limited impact on the safety case. A SCA can be considered to be maintainable when the ‘anticipated’ impact of a change can be judged to be contained within a argument module.

The modifiability sub-QC is used to assess the SCA for it ability to handle larger scale challenges to a particular safety case, e.g., challenges associated with producing safety arguments for new system functionality. A SCA can be judged to be modifiable when the anticipated effects of a large scale challenge can be limited to only the affected argument modules and the SCA maintains its own integrity. As a working hypothesis, when using the modifiability QC issues of concern would be ensuring that the argument modules are highly cohesive, i.e., contain strongly related argument and exhibit low coupling, i.e., interrelationships between argument modules are minimized.

If the results of this phase indicate that no argument suggested is acceptable or changeable enough, then it may be necessary to repeat phases 1 and 2.

Selection of SCA:

From phase 3 there should be a strong candidate for selection, if not then the evaluation and preceding phases may need repeating. Once the strongest candidate (i.e., the one that most strongly satisfies the QCs) is chosen, then reasons for making the choice, and the expected contents of the argument modules, documented. As previously mentioned in The expected contents could be linked to potential safety case patterns.

Evaluating A Safety Case Architecture

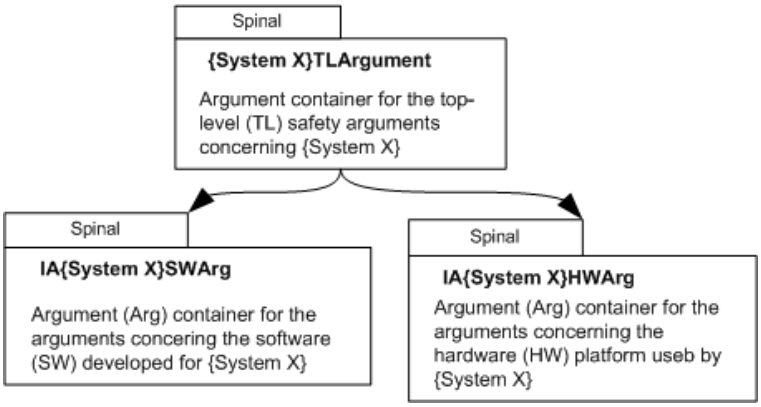

Figure 3 — A Proposed SCA

In this section the focus is on evaluating the SCA using the proposed SCA QCs. The SCA being evaluated is for an example computer based safety-critical system. As mentioned in the introduction, this system consists of software and a target hardware platform.

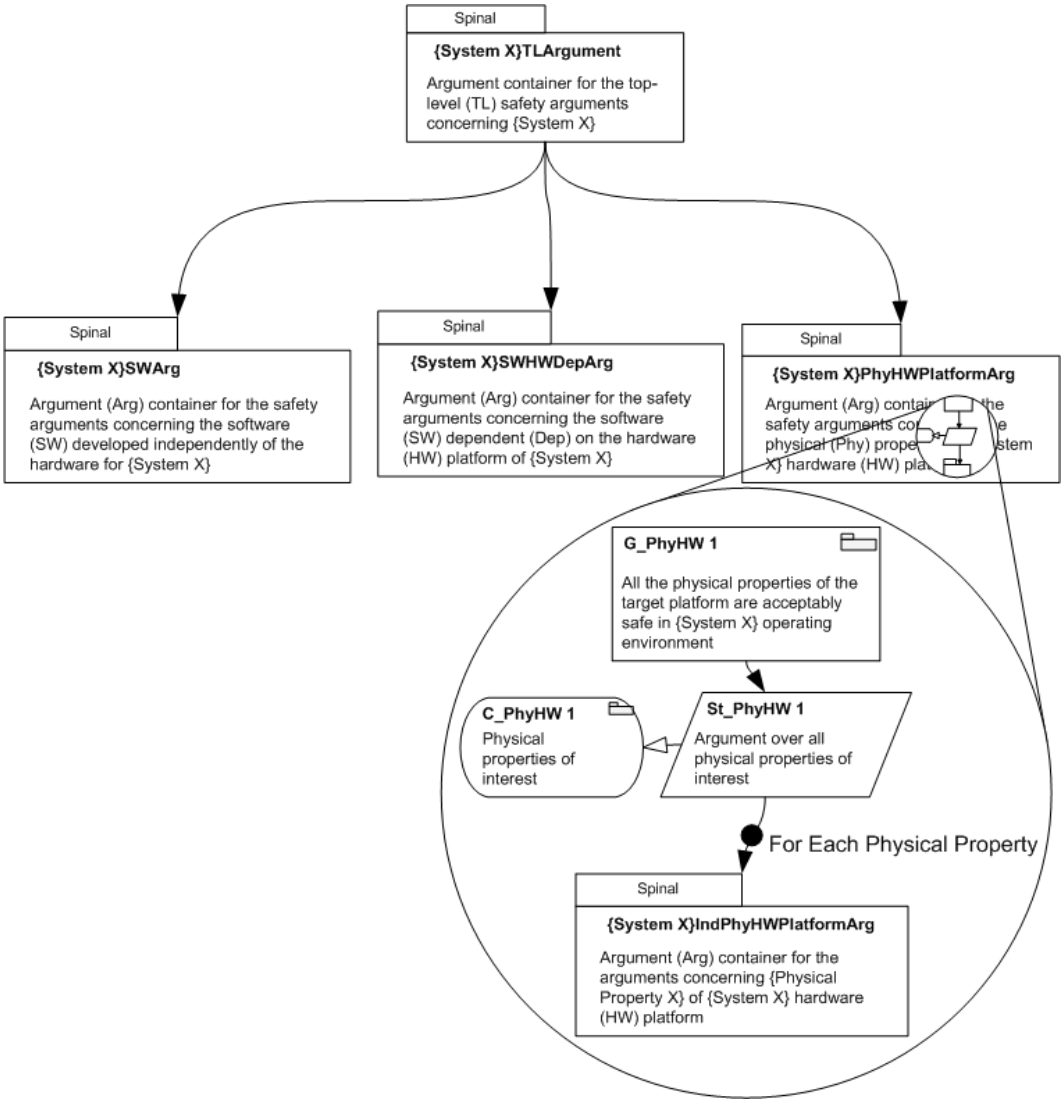

Figure 4 — The SCA Created For Computer-Based Safety-Critical System

From phases 1 and 2 (not included this paper for reasons of space) of the proposed creation process the SCA shown in figure 3 was suggested. The SCA shows approach that would divide the safety arguments into individual arguments about the software and hardware. It could be anticipated that these argument module {System X}SWArg would contain arguments about the software applications and that argument module {System X}HWArg would include arguments about, e.g, prevention of fault propagation (annex E of ref. 8).

Evaluating the proposed SCA (figure 3) using the acceptability QC reveals that the architecture is both compelling and comprehensible. Based on the system, the SCA could be judged to be compelling because the anticipated arguments could be both plausible and expected to cover all aspects of the system. The arguments could be judged to be plausible because, e.g., several existing standards and guidance documents divide the system into hardware and software, i.e., UK MoD safety standards 00-56 (System), 00-55 (software) , and 00-54 (hardware), and FAA guidance documents DO178B (software) and DO254 (hardware). The proposed SCA could be judged to comprehensible because it allows a user to comprehend how the arguments contained in the argument modules relate to each other. Furthermore, the arguments are unambiguous (identified as an aspect of comprehensibility) because it is clear what the argument modules scope and contribution to the modular safety case would be.

Evaluating the proposed SCA using the changeability QC reveals the SCA is maintainable but not modifiable. The maintainability QC indicates that the proposed SCA would make it easier to locate the parts of the safety case that would be affected by changes to the hardware platform or to the software, based on a comparison to a traditional safety case. However, the modifiability characteristic revealed that the potential for cross-linkages between the argument modules could be significant and would probably not be suitable to support the long time use of the SCA.

The evaluation for modifiability led to the revisiting phase 2 of the proposed creation process. The result was the refinement shown in figure 4. From the figure it can be seen that the argument modules could be anticipated to contain arguments that are closely related to either the software (i.e., {System X}SWArg) or the hardware ({System X}PhyHWArg), and argument that are closely related to both software and hardware (i.e., {System X}SWHWDepArg).

Evaluation using the QCs this time indicated that the SCA shown in figure 4 satisfied all the QCs. In particular, the SCA could be judged to be sufficiently modifiable as it could now be possible to change the hardware without affecting the arguments about the software and vice versa. For example, in this SCA the:

- {System X}SWArg is anticipated to contain arguments about the software's contribution to system level hazards,
- {System X}SWHWDepArg is anticipated to contain arguments about acceptable run-time properties, acceptable compiler integrity, etc.,
- {System X}PhyHWArg is anticipated to contain arguments about the acceptability of the target hardware platform's physical properties (as shown by the GSN in the circle in figure 4).

Thus, when a change is made to the target hardware platform, assuming the new hardware platform does not introduce new system level hazards to which the software contributes, it could be predicted that the change will strongly impact the {System X}PhyHWArg and weakly affect parts of the {System X}SWHWDepArg and not impact the {System X}SWArg. Hence, the change will have been isolated to parts of the SCA, and not require changes to all the SCA. Furthermore, easier location of the change will have been facilitated

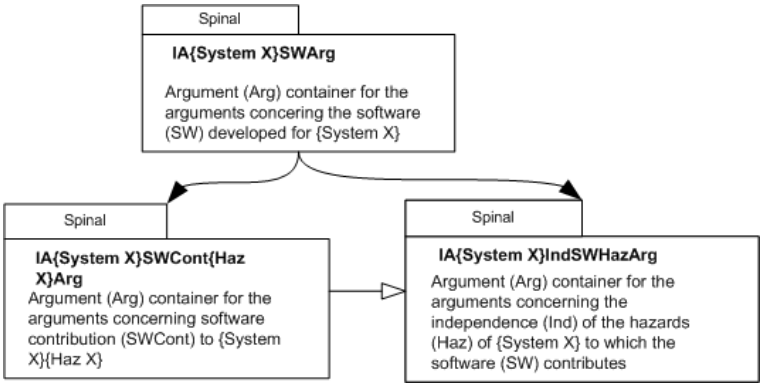

Figure 5 — Possible Further Decomposition of {System X}SWArg

The final stage to the evaluation phase is to ask if any further decomposition would add value or new concepts. This stage is identified because there is a point at which the SCA provides enough information to the users such that it can be used for their own purposes and further decomposition would add no value or concepts. For example, value and concepts may be added by further decomposition to the {System X}SWArg such as decomposition into argument modules about the individual system hazards that the software may contribute to and an argument module about the independence of the hazards (fig. 5). At this stage the acceptability QC is most applicable to making this judgement. For example, does the decomposition shown in figure 5 add value and concepts, or does the anticipated scope of the {System X}SWArg (i.e., that it contains arguments about the software's contribution to system level hazards) provide enough information to SCA users. This judgement is up to SCA developers and guidance for making this decision will derived from the continuing investigation into SCAs.

Conclusions and Future Work

In this paper we have defined a SCA as the master plan for safety case and have strongly implied that a SCA should be used to create a modular safety case. Modular safety cases are safety cases that are decomposed into argument modules. It was briefly shown how a SCA facilitate the construction of a modular safety case and how the modular features of GSN, a widely used argument notation, can be used for their representation.

In this paper we introduced a four phase iterative process for creating SCAs and explained the evaluation through an example SCA. In this paper the SCA was based on a computer based safety-critical system. Two QCs (acceptability and changeability) were introduced for the evaluation of a SCA. Future work on QCs will attempt to rationalize the results of an investigation of their use into heuristics and questions to aid the creation and evaluation of a SCA. Furthermore, future work will investigate imaginative anticipation and the use of safety case patterns to improve the creation process.

In this paper we proposed a process for creating and evaluating SCA was presented. The future work will investigate how a SCA can be used will investigate how a SCA can be used to improve change management. Other future work will investigate documenting SCAs to provide enough information to support their through-life usage.

References

1. Barry W. Boehm, John R. Brown, Hans Kaspar, Myron Lipow, Gordon J.Macleod, and Michael J. Merrit. Characteristics of Software Quality. North-Holland Publishing Company, 1978.

2. Mark Dowding. Maintenance of the certification basis for a distributed control system - developing a safety case architecture. Master's thesis, The University of York, 2002.

3. Tim P. Kelly. Arguing Safety - A Systematic Approach to Managing Safety Cases. PhD thesis, The University of York, September 1999. YCST 1999/05.

4. Tim P. Kelly. Concepts and principles of compositional safety case construction. Technical Report COSMA/2001/1/1, The University of York, 2001.

5. Tim P. Kelly. Managing complex safety cases. In Felix Redmill and Tom Anderson, editors, Current Issues in Safety-Critical Systems, pages 163–172. Springer-Verlag, 2003.

6. Jon Lang. Creating Architectural Theory: The Role of the Behavioural Sciences in Environmental Design. John Wiley & Sons Inc, 1987.

7. MOD. Defence Standard 00–56(Part 2)/Issue 2 – Safety Management Requirements for Defence Systems (Part 1: Guidance). Ministry of Defence, December 1996.

8. MOD. Defence Standard 00–54(Part 2)/Issue 2 – Requirements For Safety Related Electronic Hardware in Defence Equipment (Part 2: Guidance). Ministry of Defence, March 1999.

9. John Rushby. Modular certification. Technical Report NASA/CR-2002-212130, Langley Research Center, 2002.

10. D. N. Walton. Argumentation and theory of evidence. New Trends in Criminal Investigation and Evidence, Volume II:711–732, 2000.

11. Robert Weaver, John McDermid, and Tim Kelly. Software safety arguments: Towards a systematic categorisation of evidence. In Proceedings of the 20th International System Safety Conference, pages 812–821, August 2002.

Biography

Simon Bates has been a Research Associate in the BAE SYSTEMS funded Dependable Computing Systems Centre (DCSC) at the University of York since October 2002. He graduated from the University of Manchester in 2002, where he attained a MEng (Hons) in Electronic Systems Engineering. Since taking up his role with the DCSC he

has developed the following research interests: Safety Cases, Safety Case Architectures, Modular and Incremental Certification of Safety Related Systems, and Software Architectures.

Dr Iain Bate has been a Researcher within the Real-Time Systems Research Group within the Department of Computer Science at the University of York since 1994. His research has been in the area of real-time systems, systems engineering, code generation, analysis techniques, architecture assessment and optimisation, and integrated modular avionics related to safety-critical systems. His doctoral research, completed in 1998, focussed upon establishing and demonstrating an approach to scheduling and timing analysis for safety-critical systems. The approach has been adopted by Rolls-Royce for use on aircraft projects.

John McDermid has been Professor of Software Engineering at the University of York since 1987 where he runs the high integrity systems engineering (HISE) research group. HISE studies a broad range of issues in systems, software and safety engineering, and works closely with the UK aerospace industry. Professor McDermid is the Director of the Rolls-Royce funded University Technology Centre (UTC) in Systems and Software Engineering and the BAE SYSTEMS-funded Dependable Computing System Centre (DCSC). He is author or editor of 6 books, and has published about 250 papers.